Problem 11292: Dragon of Loowater

Source Code

Author: Chua, Kyle Matthew C.

```
#include <stdio.h>
#include <stdlib.h>

int compare (const void * a, const void * b)
{
   return ( *(int*)a - *(int*)b );
}

int main(){
      int n, m;
      int heads[20000];
      int knights[20000];
      int i;
      int j;
      int ctr;
      int cost;
      scanf("%d %d", &n, &m);
      while(n != 0 && m != 0){
            ctr = 0;
            cost = 0;
            for(i=0; i<n; i++)
                  scanf("%d", &heads[i]);
            for(i=0; i<m; i++)
                  scanf("%d", &knights[i]);
            if(n > m)
                  printf("Loowater is doomed!\n");
```

```
            else{
                  qsort(heads, n, sizeof(int), compare);
                  qsort(knights, m, sizeof(int), compare);
                  while(heads[0] > knights[ctr])
                        ctr++;
                  if(n > m-ctr)
                        printf("Loowater is doomed!\n");
                  else{
                        j = ctr;
                        i = 0;
                        while(i < n && j < m){
                              if(heads[i] > knights[j])
                                    j++;
                              else{
                                    cost += knights[j];
                                    j++;
                                    i++;
                              }
                        }
                        if(i == n)
                              printf("%d\n", cost);
                        else
                              printf("Loowater is doomed!\n");
                  }
            }
            scanf("%d %d", &n, &m);
      }
      return 0;
}
```

Discussion:

To minimize the cost of slaying the dragon, the approach used in the algorithm was greedy method. Sorting the head's size and the knight's cost in ascending order is very important in the algorithm. The function compare will be used as one of the parameters of the qsort() function in the program which will be used later on. The program flow is shown in the discussion below:

1. Ask the user for 2 inputs, one for the head count and the other for knight count.
2. If input is not 0 and 0, the program proceeds otherwise end the program.
3. The program will ask the user to input a set of values depending on the specified number of heads and knights. The input values for the head size will be stored in an array as well as the height or cost of each knight in a different array.
4. Check if the number of heads is greater than the number of knights, if yes the program will display "Loowater is doomed!" and go back to step 1, otherwise the program proceeds.
5. Sort both arrays in ascending order using qsort() (quicksort) function.
6. Check if the number of heads is greater than the number of knights that can chop the smallest head of the dragon denoted by heads[0], if yes the program will display "Loowater is doomed!" and go back to step 1, otherwise the program proceeds.
7. We then iterate all the heads of the dragon or until there are no more knights left and get the knights that can chop a head of the dragon staring with the knight that can chop the smallest head of the dragon denoted by knight[ctr]. If the knight cannot chop the current head of the dragon we are checking, check the next knight, otherwise we add the knight's expense to the cost variable and check the next head.
8. If variable i is equal to variable n, denoting that we have the resources to chop all the heads of the dragon, we output the cost to do so. Otherwise Loowater is doomed!
9. Return to step 1

Time Complexity:

```
#include <stdio.h>
#include <stdlib.h>

int compare (const void * a, const void * b)
{
   return ( *(int*)a - *(int*)b );
}

int main(){
      int n, m;
      int heads[20000];
      int knights[20000];
      int i;
      int j;
      int ctr;
      int cost;
      scanf("%d %d", &n, &m);
      while(n != 0 && m != 0){
            ctr = 0;
            cost = 0;
            for(i=0; i<n; i++)
                  scanf("%d", &heads[i]);
            for(i=0; i<m; i++)
                  scanf("%d", &knights[i]);
```

O(1)

O(1)

Ignored because it only checks if the program should continue or not.

O(n)

```
        if(n > m)
                printf("Loowater is doomed!\n");
        else{
                qsort(heads, n, sizeof(int), compare);
                qsort(knights, m, sizeof(int), compare);
                while(heads[0] > knights[ctr])
                        ctr++;
                if(n > m-ctr)
                        printf("Loowater is doomed!\n");
                else{
                        j = ctr;
                        i = 0;
                        while(i < n && j < m){
                                if(heads[i] > knights[j])
                                        j++;
                                else{
                                        cost += knights[j];
                                        j++;
                                        i++;
                                }
                        }
                        if(i == n)
                                printf("%d\n", cost);
                        else
                                printf("Loowater is doomed!\n");
                }
        }
        scanf("%d %d", &n, &m);
    }
    return 0;
}
```

O(1)

O(nlogn) or $O(n^2)$

O(nlogn) or $O(n^2)$

O(1)

O(n)

O(n)

O(1)

O(1)

O(n)

O(1)

O(1)

O(1)

O(1)

O(1)

O(nlogn) or $O(n^2)$

O(1)

Total: **O(nlogn) or $O(n^2)$ depending on quicksort efficiency.**

References:

- http://www.tutorialspoint.com/c_standard_library/c_function_qsort.htm